From Montague Grammar to Database Semantics

Roland Hausser* †
Universität Erlangen-Nürnberg(em)

Roland Hausser. 2015. From Montague Grammar to Database Semantics. *Language and Information 19.2, 1-18.* This paper retraces the development of Database Semantics (DBS) from its beginnings in Montague grammar. It describes the changes over the course of four decades and explains why they were seen to be necessary. DBS was designed to answer the central theoretical question for building a talking robot: How does the mechanism of natural language communication work? For doing what is requested and reporting what is going on, a talking robot requires not only language but also non-language cognition. The contents of non-language cognition are re-used as the meanings of the language surfaces. Robot-externally, DBS handles the language-based transfer of content by using nothing but modality-dependent unanalyzed external surfaces such as sound shapes or dots on paper, produced in the speak mode and recognized n the hear mode. Robot-internally, DBS reconstructs cognition by integrating linguistic notions like functor-argument and coordination, philosophical notions like concept-, pointer-, and baptism-based reference, and notions of computer science like input-output, interface, data structure, algorithm, database schema, and functional flow. [1] (**Universität Erlangen-Nürnberg**)

Key words: human-machine communication, turn-taking, time-linear derivation order

The starting point in the development of DBS was the linguistic study content taught at university [2], here Montague grammar. Compared to the dominant schools of nativism, Montague grammar was shown preferable because it offered a formal representation of meaning in the form of truth conditions defined relative to a set-theoretic model structure.

Seemingly the only well-defined formal semantics possible, truth-conditional semantics has since been absorbed into the linguistic main stream. What more could one wish for than a

---

* This is an invited paper of *Language and Information 19.2*.

† Universität Erlangen-Nürnberg(em), rrh@linguistik.uni-erlangen.de

1 For a brief presentation of DBS from a functional point of view see Hausser (2013).

2 Thanks to Helmut Schnelle, TU Berlin (1968–1970), and to Stanley Peters, UT Austin (1970–1974).

formal grammar of a natural language with a semantic interpretation? The real question, however, is whether such a system allows to reconstruct the mechanism of free language communication, verified in the form of a talking robot.

Designed long before the advent of computers, truth-conditional semantics is *sign-oriented*: it uses a metalanguage to define a set-theoretic model which contains the language signs and the world, and defines reference as a direct relation between the signs and the artificial world. A talking robot, in contrast, requires an *agent-oriented* approach. The real world is treated as given; the robot interacts with it via its external interfaces and internal cognition.

1. Ontology

The dichotomy between a sign- and an agent-oriented approach to natural language is a question of ontology, in the sense of philosophy.[3] To arrive at an agent-oriented approach, the metalanguage-based relation between language and "the world" of the sign-oriented approach must be replaced with an agent-internal reference relation between cognitive representations of the language meaning and the context of use (CLaTR Sect. 4.3).

To accommodate truth-conditional semantics in an agent-oriented approach, SCG proposed to combine two set-theoretic models inside the agent, one representing the language meaning, the other the context of interpretation. The resulting [+sense, +constructive] ontology (FoCL Sect. 20.4) accommodates the First Principle of Pragmatics (PoP-1, NLC 2.6.1) and the Principle of Surface Compositionality (NLC 1.6.1), at least conceptually.

Given the opportunity to use the computers at CSLI Stanford (1984-86), we attempted to computationally verify the SCG fragment of English with an implementation in Lisp. This seemed possible for the following reasons:

1.1 Why programming the SCG fragment seemed possible

1. The SCG fragment is strictly formalized in accordance with Montague grammar, widely considered the highest standard of formal explicitness in truth-conditional semantics.
2. In accordance with PoP-1, the syntactic-semantic SCG analysis disregards the possible meaning$_2$ interpretations which may result from different uses of an expression relative to different contexts of interpretation.
3. Transformations and other operations known to increase complexity to exponential or

3 Thanks to Nuel Belnap and Rich Thomason at the Philosophy Department of the University of Pittsburgh (1978–1979), and to Julius Moravscik and Georg Kreisel at the Philosophy Department of Stanford University (1979–1980, 1983–1984) for revealing the secrets of a Tarskian semantics. Later, at Carnegie Mellon University (1986–1989), understanding this fundamental topic was helped further by Dana Scott (FoCL Chaps. 19–21). The 1978–1979 stay in Pittsburgh and the 1979–1980 stay at Stanford were supported by a two-year DFG research grant.

undecidable are excluded from the syntactic-semantic analysis of the language signs by the Principle of Surface Compositionality.

Unfortunately, however, the SCG fragment turned out to be seriously unsuitable for a computational implementation.

The immediate problem was the formalism of categorial grammar (C grammar), which is part and parcel of Montague grammar. Designed by Leśniewski (1929) and Ajdukiewicz (1935), the combinatorics of C grammar are coded into lexical categories, using only two canceling rules in a nondeterministic bottom-up derivation order (FoCL Sect. 7.4). This is elegant and simple intuitively, but the derivation order is underspecified by the algorithm and relies heavily on human intelligence (FoCL Sect. 7.5).

More specifically, it is not obvious whether or not there exists a correct initial composition somewhere in middle of the input chain and if so, where; this has the character of problem solving even for experts. Also, C Grammar requires a high degree of lexical ambiguity for coding alternative word orders into alternative categories (FoCL Sect. 7.6). Therefore, a computer program based on a C grammar for a natural language requires inordinate amounts of trial and error, which made programming the SCG fragment impossible.[4]

The experience confirmed the more general insight that a rigorous formalization is a necessary, but not a sufficient, condition for ensuring a reasonable implementation as a well-designed software. What is needed in addition is the declarative specification of the interfaces, the data structure, the components, the database schema, the functional flow, and the motor driving the derivation, based on a general idea of how natural language communication works.

The theory of natural language communication must respect the obvious structures of human cognition. For example, long-term upscaling will fail if the timelinear structure of language expressions is (mis)treated as the "problem of serialization." It will also fail if the agent's external input and output interfaces, it's memory, and the algorithm mapping between the two are ignored. This is confirmed by the long-term upscaling failures of today's main stream theories of language.

2. Functional Flow

The human prototype uses its external recognition interfaces to take a time-linear sequence of modality-dependent unanalyzed external surfaces as input, maps them into content which is stored and processed in memory, and realized by the agent's external action interfaces as

4 Even today, no implementation of a C grammar exists for any fragment of a natural language with nontrivial data coverage. Such an implementation would be useful for working with this formal theory. Without it, C grammars have found no direct practical applications in their long history.

raw output. This cannot be modeled by a C grammar because, instead of a time-linear derivation order, the start of the derivation is usually somewhere in the middle of the input. Thus, the second reason why NEWCAT had to abandon the Montague grammar defined in SCG is the inadequate functional flow of C grammar.

C grammar is motivated by hierarchical constituent structures, which are also used by phrase structure grammar (PS grammar).[5] Based on the principle of possible substitutions,[6] context-free PS grammar generates different expressions from a single node, usually called S – without any external interfaces.

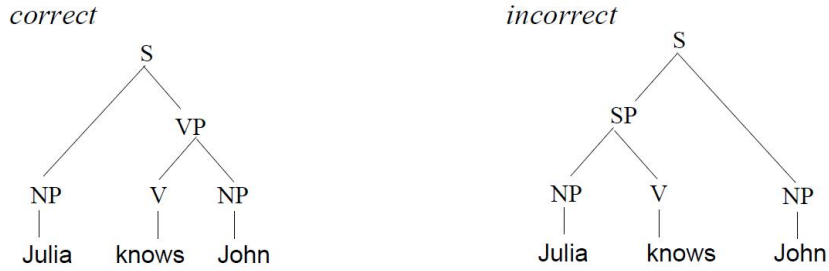
Formally, constituent structures are defined in terms of context-free phrase structure trees which fulfill the following conditions:

2.1 Definition of constituent structure

1. Words or constituents which belong together semantically must be dominated directly and exhaustively by a node.
2. The lines of a constituent structure may not cross (non-tangling condition).

According to this definition, the first of the following two phrase structure trees is a linguistically correct analysis, while the second is not:

2.2 Correct and incorrect constituent structure analysis



There is general agreement among nativists that the words knows and John belong closer together semantically than the words Julia and knows.[7] Therefore, only the tree on the left is considered grammatically correct.

On the one hand, from a formal point of view both phrase structure trees are equally

5 As proven by C. Gaifman in June 1959, bidirectional C grammar and context-free PS grammar are weakly equivalent (Bar Hillel 1964, p. 103). See also Buszkowski (1988) and FoCL Sect. 9.2.

6 PS grammar uses the principle top down, while C grammar uses it bottom up (FoCL 10.1.6).

7 To someone not steeped in nativist linguistics, these intuitions may be difficult to follow. They are related to the substitution and movement tests of American structuralism (FoCL Chap. 8).

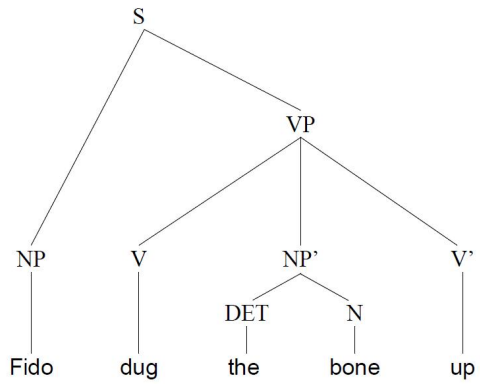
well-formed. On the other hand, the number of formally possible trees grows exponentially with the length of the sentence.[8] Such a large number of wellformed phrase structure trees for one and the same sentence would be meaningless descriptively if they were all linguistically correct.

Therefore the linguistic concept of constituent structure as defined in 2.1 is crucial for any phrase structure analysis, be it in PS grammar or in C grammar: constituent structure is the only intuitive principle[9] widely accepted in nativism for excluding most of the formally possible trees.

Yet it has been known at least since 1953 (Bar-Hillel 1964, p. 102) that there are certain natural language constructions, called "discontinuous elements," which violate the definition of constituent structure. In other words, constituent structure fails to always fit the data. Thus, the third reason why the Montague grammar defined in SCG had to be abandoned is the empirical inadequacy of the constituent structure common to PS and C grammar.

As an example, consider two attempts at defining a constituent structure for Fido dug the bone up, containing the discontinuous element dug__up:

2.3 First attempt: violating condition 1



The lines do not cross, satisfying the second condition of 2.1. The analysis violates the first condition, however, because the semantically related expressions dug__up, or rather the nodes V (verb) and V' (discontinuous element) dominating them, are not dominated *exhaustively* by a node. Instead, the VP node directly dominating V and V' also dominates

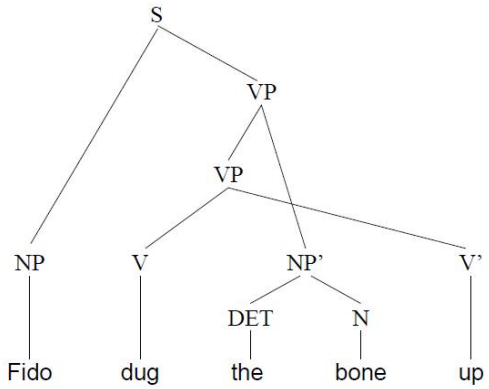
8 If loops like A → ...A... are permitted in the rewrite rules (which they usually are), the number of different possible trees over a finite sentence is infinite!

9 Historically, the definition of constituent structure is fairly recent; it goes back to the *immediate constituent* (IC) analysis of Bloomfield (1933).

the NP' the bone.

The other attempt is as follows:

2.4 Second attempt: violating condition 2



Here the semantically related subexpressions dug and up are dominated directly and exhaustively by a node, thus satisfying the first condition of definition 2.1. The analysis violates the second condition, however, because the lines in the tree cross.

Discontinuous elements are a problem because constituent structure analysis is defined in terms of context-free PS grammar. This class generates pairwise inverse relations, e.g. abc...cba, and is of polynomial complexity. Relations which are not pairwise inverse, such as abc...abc (not inverse, FoCL 11.5.6) and aaabbbccc (not pairwise, FoCL 10.2.3), are at least context-sensitive. This class is exponential and thus computationally intractible.[10]

The distinction between context-free and context-sensitive relations is an artefact of the PS grammar rule formats (FoCL 8.1.2) and has no foundation in natural language. Given the empirical inadequacy of PS grammar for computational linguistics, we had no choice but to find a new, more suitable algorithm. Called LA grammar, it parses all of the above relations in linear time (TCS) and discontinuous elements do not increase complexity (3.3).

In natural language, discontinuous structures are not a marginal phenomenon. They occur frequently, both within a language and in the different languages of the world. For example, an ubiquitous discontinuous construction in German is the perfect tense of the transitive verb in declarative main clauses, as in Peter hat das Buch gelesen, with discontinuous hat__gelesen.

LA grammar avoids the Constituent Structure Paradox by replacing the hierarchical

10 The complexity of transformational grammar is even higher than context-sensitive, namely undecidable (Peters and Ritchie 1973).

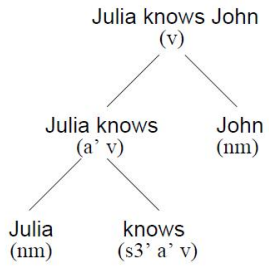
structure defined in 2.1 with a more fundamental principle, namely the timelinear structure of natural language – in accordance with de Saussure's 1913/1972 second law (*principe seconde*). Time-linear means linear like time and in the direction of time (in contradistinction to linear time, which is a complexity degree). In DBS, the time-linear structure of natural language is realized in the derivation order[11] of the hear, the think, and the speak mode.

3. Computing Possible Continuations

The problems encountered in the attempt to program the Montague grammar defined in SCG motivated the design of the Lisp program published in NEWCAT. It demonstrated that a strictly time-linear analysis of natural language may be simple, efficient, and linguistically well-motivated in terms of the functor-argument and coordination structures at the elementary, the phrasal, and the clausal level. The scope of the fragment were 221 grammatical constructions of German and of 114 grammatical constructions of English. That the software and the explaining text of NEWCAT could be written in six months may be taken as additional support for a time-linear approach to the linguistic analysis of natural language.

The following reanalysis of example 2.2 replaces the underspecified derivation order of substitution-based grammars with the strictly time-linear derivation order of continuation-based LA grammar (to be read bottom-up):

3.1 Conceptual NEWCAT analysis of Julia knows John



The time-linear analysis combines a *sentence start* and a *next word* into a new sentence start.

11 Nativism treats derivation order as a "performance" phenomenon, irrelevant for "competence." The reason is the partial order of possible substitutions, as reflected in the many possible derivation orders for parsing phrase structure trees, like left-corner, right-corner, island, etc. Possible substitutions express the nativist view of grammar as a *generation* mechanism, like describing the growth of a plant, and not as a mechanism for the transfer of content between agents.

The combination is based on valency canceling (Dependency grammar, Tesnière 1959). For example, the sentence start (Julia (nm)) and the next word (knows (s3′ a′ v)) are combined into the new sentence start (Julia knows (a′ v)). The category segment nm (for name) serves as a valency filler which cancels the valency position s3′ in the category of knows. The time-linear procedure continues until (i) there is no more next word available in the input or (ii) an ungrammatical continuation is encountered.

In NEWCAT, the computation of possible continuation serves as the automatic grammatical analysis: the software operations are displayed as a suitably formatted *trace* in the sense of computer science. In this way, absolute 'type transparency' is achieved (FoCL Sect. 9.3; Berwick and Weinberg 1984):

3.2 Automatic NEWCAT parse of example 3.1

```
NEWCAT>  Julia knows John \.

     *START
     1
        (SNP) JULIA
        (N A V) KNOWS
     *NOM+FVERB
     2
        (A V) JULIA KNOWS
        (SNP) JOHN
     *FVERB+MAIN
     3
        (V) JULIA KNOWS JOHN
        (V DECL) .
     4
     *CMPLT
        (DECL) JULIA KNOWS JOHN .
```

The comparatively vague intuitions about what "belongs semantically together" (which underlie the definition of constituent structure 2.1) are replaced by the semantic relations of functor-argument and coordination, and coded in categories defined as lists of one or more category segments. In this way, the constituent structure paradox is avoided, as shown by the following NEWCAT parse:

3.3 NEWCAT parsing of Fido dug the bone up

```
NEWCAT>  Fido dug the bone up \.

    *START
    1
       (SNP) FIDO
       (N A UP V) DUG
    *NOM+FVERB
    2
       (A UP V) FIDO DUG
       (SN SNP) THE
    *FVERB+MAIN
    3
       (SN UP V) FIDO DUG THE
       (SN) BONE
    *DET+NOUN
    4
       (UP V) FIDO DUG THE BONE
       (UP) UP
    *FVERB+MAIN
    5
       (V) FIDO DUG THE BONE UP
       (V DECL) .
    *CMPLT
    6
       (DECL) FIDO DUG THE BONE UP .
```

The discontinuous element up is treated as a filler for the valency position up′ in the lexical category (n′ a′ up′ v) of dug.[12] The phrasal noun the bone is added in two time-linear steps: the article the has the category (sn′ snp) such that the category segment snp cancels the valency position a′ in the category (a′ up′ v) of Fido dug, while the category segment sn′ is added in the result category (sn′ up′ v) of Fido dug the. In this way the obligatory addition of a noun after adding the determiner the in English is ensured.

The NEWCAT Lisp source code was re-implemented by several readers in South America, Switzerland, and other locations.[13] Later, an algebraic definition[14] was distilled from the Lisp code (CoL, TCS).

---

12 The use of ′ to distinguish a valency position, e.g. a′, from a valency filler, e.g. a, was introduced later. The proper treatment of grammatical number in definite noun phrases (Choe et al. 2006) had not yet been found.

13 These efforts became known because one little function had been omitted accidentally in the NEWCAT source code publication, causing several of the re-programmers to write and ask for it.

14 Thanks to Stuart Shieber and Dana Scott, who at different times and places helped in formulating the algebraic definition for LA grammar. The author's 1983–1986 stay at Stanford University and the subsequent 1986–1988 stay at Carnegie Mellon University were supported by a five-year DFG Heisenberg grant. The 1988–1989 stay at CMU was supported by a Research Scientist position at the LCL (Dana Scott and David Evans). Thanks also to BrianMacWhinney, who supported a fruitful three-month stay at the CMU Psychology Department in the fall of 1989.

4. Deriving Content

The design of the NEWCAT parser solved three problems. First, by replacing the underspecified derivation orders of C and PS grammar with a strictly time-linear derivation order, we arrived at the new algorithm of LA grammar, which parses the natural languages in linear time (FoCL Sects. 12.5 and 21.5). Second, the time-linear building up and canceling of valency positions provides a semantically motivated analysis of natural language which eliminates the Constituent Structure Paradox (Sect. 2). Third, the failure of C and PS grammar to be I/O equivalent with the human prototype is repaired by defining the algorithm of NEWCAT as a time-linear LA hear grammar which takes a sequence of unanalyzed word form surfaces as input.

What was missing, however, was a derivation of *content* in the hear mode, to be processed in the think mode, and used as input to the speak mode. As in C grammar, NEWCAT derivations of different declarative sentences all end in the same category, namely (v),[15] i.e., a verb without any unfilled valency positions.[16] The question was how to provide the NEWCAT parsing of natural language with a semantic representation of (i) language content ($meaning_1$) which is suitable also for building (ii) the context of interpretation, as a structural precondition for (iii) a simple pattern matching between the language content and the context content in accordance with PoP-1 (NLC 2.6.1).

The most main stream solution at the time would have been a truth-conditional approach. However, while attempting to program the SCG fragment at CSLI Stanford in 1984, it had already become clear that two set-theoretical models (one representing the $meaning_1$, the other the context of use) are not suitable[17] for a computationally viable pattern matching (FoCL 22.2.1).

The perhaps second most main stream solution at the time was in the form of phrase structure trees, interpreted as semantic hierarchies. This was attempted in CoL. Proceeding along the then popular separation of syntax and semantics ("autonomy of syntax"), the basic idea was to use the NEWCAT parser as a syntactic algorithm and define a semantic interpretation for it.[18]

For complementing NEWCAT derivations with a parallel construction of semantic hierarchies we used the FrameKit+ software (Carbonell and Joseph 1986). The resulting CoL

15 Assuming that the period is omitted (3.2, 3.3).

16 This is similar to Montague grammar, in which the derivation of different constructions all end in the category t (for true) if successful.

17 Thus, even if a C grammar with lambda reduction were to derive set-theoretical models from language expressions in a time-linear derivation order (as proposed by Kempson et al. 2001), these models would not be practical for a [+sense, +constructive] (FoCL 20.4.2) reconstruction of reference in natural language communication.

18 Thanks to Jaime Carbonell and the CMT/LTI at Carnegie Mellon University (1986–1989), who made programming the semantically interpreted CoL parser possible.

software provided the automatic NEWCAT analyses, extended to 421 grammatical constructions of English, with homomorphic semantic hierarchies. The semantic interpretation of 3.3 is as follows (CoL, p. 44):

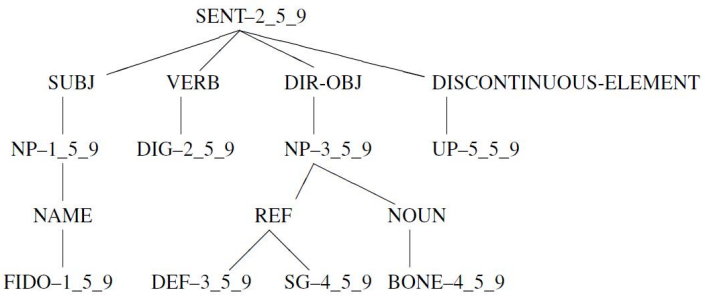
4.1 Semantic interpretation as a frame structure in CoL

```
Hierarchical Analysis:

(SENT-2_5_9
 (SUBJ ((NP-1_5_9 (NAME (FIDO-1_5_9)))))
 (VERB (DIG-2_5_9))
 (DIR-OBJ ((NP-3_5_9 (REF (DEF-3_5_9 SG-4_5_9))
                                    (NOUN ((BONE-4_5_9))))))
 (DISCONTINUOUS-ELEMENT ((UP-5_5_9))))
```

Using a suitable tree-printer, this structured list may be automatically displayed as the following semantic hierarchy (CoL, p. 45):[19]

4.2 Displaying the frame structure 4.1 as a tree



The time-linear syntactic analysis 3.3 and its semantic interpretation 4.1 are derived simultaneously.

Superficially, 4.2 may seem to resemble a constituent structure, but neither its intuitive assumptions nor its formal definition 2.1 are satisfied. In particular, the assumption that dug is semantically closer to the bone than to the dog.[20] is not expressed. Nota bene: all constituent structures are semantic hierarchies, but not all semantic hierarchies are constituent structures.

---

19 The tree printer available at the time connects nodes with the corners ⌜ and ⌝ in typewriter font. For better readability, the corners have been replaced here by the diagonal lines familiar from PS trees.

20 In context-free PS grammar, this assumption is formally expressed by the rules S→NP VP and VP → V NP. DBS, in contrast, follows the logical tradition by treating the subject and a possible object as equal arguments, as in f(a, b).

Having shown how to automatically supply a substantial fragment of a timelinear NEWCAT syntax with a surface compositional, homomorphic semantic interpretation of a traditional kind, there arose the question of what to do with it. Extending the fragment for bigger tree banks or for linguistic analysis in corpus linguistics was not tempting.

Instead, we tried to turn CoL into an agent-oriented system. In a first step, we attempted to treat reference as a pattern matching between two frame structures, one representing the meaning1, the other the context of interpretation. It turned out, however, that frames – like set-theoretical models – are inherently unsuited for defining a software-mechanical pattern matching (FoCL 22.2.2):

4.3 Basic problems for the matching of frame structures

1. A content may be coded as different, but equivalent, frame structures; such variations[21] obstruct pattern matching when it should be possible.
2. The recursive embedding of frames (4.1), used for establishing semantic relations, complicates a computationally viable matching.

Problem (1) was solved by representing a content as an *order-free*[22] set of proplets. Instead of connecting the elements of a proposition by their position in a formula, behind a quantifier, in a tree, or in a frame, the proplets of a proposition are connected by a common proposition number.

Problem (2) was solved by defining proplets as *non-recursive* feature structures. Instead of representing functor-argument structure by embedding the feature structure of the argument(s) into the feature structure of the functor,[23] as in nativism, the semantic relations of functor-argument and coordination are coded alike as proplet-internal addresses and implemented as pointers.

The matching between a pattern proplet and a content proplet is straightforward and efficient because of the fixed order of attributes and the use of flat features. The matching of values is based on (i) types matching tokens (NLC, Sect. 4.1), and on (ii) variables matching constants (CLaTR Sects. 3.2, 4.3).

Pattern matching is the basic mechanism of DBS. It is used (i) in reference as a matching

---

21 Consider the propositional contents (p∨q) and (q∨p). Because they are semantically equivalent, they should match, but their structural difference would require extra work. In DBS, this proplem is solved by recoding propositional calculus for natural language as order-free sets of proplets which specify inter-proplet relations in the form of addresses (Hausser 2003).

22 The assumption of an order-free content seems to agree with the SemR level of the Meaning-Text theory (MT) proposed by Zholkovskij and Mel'chuk (1965). An order-free level is also used in some dependency grammars, e.g., Hajič̆cová (2000).

23 Recursive embedding cannot even be properly extended to coordination – which is the other basic semantic relation of structure in Aristotelian semantics besides functor-argument.

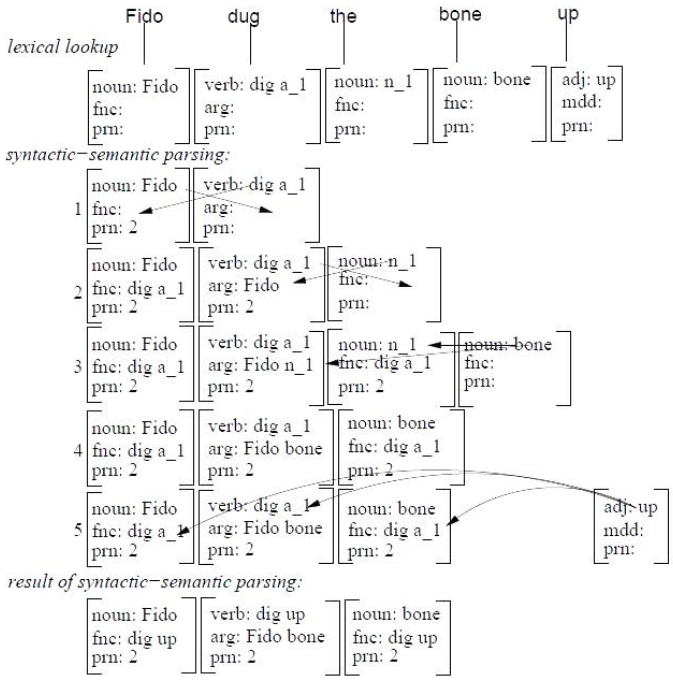
between a language meaning1 and a context of interpretation (NLC2 3.2.4), (ii) in the recognition and action procedures of peripheral cognition (NLC 4.5.2), and (iii) in applying the operations of inferencing and the hear, think, and speak mode derivations.

5. Graphical Representations in DBS

For theoretical and practical reasons, we developed DBS and the SLIM theory on which it is based (NLC Sect. 2.6) as a *declarative specification* in the sense of computer science (NLC 1.2.1). A declarative specification may be defined in terms of formal rules, for example, a logical production system or an LA grammar. A sometimes more intuitive approach, however, is graphical. This holds especially for the characterization of the component structure and the functional flow in a complex system. It also holds for characterizing the derivations in the hear, the think, and the speak modes of DBS.

Compared to representing a hear mode derivation as a trace of the NEWCAT parser (3.2, 3.3), the NLC format is more comprehensive, based on the data structure of proplets. Consider the graphical format of NLC in the hear mode derivation of our discontinuous element example:

5.1 NLC hear mode derivation of Fido dug the bone up



The format resembles 3.3 in that both are to be read top-down – in contradistinction to 3.1 which is to be read bottom-up (like C grammar trees).

The earlier NEWCAT derivations run on categorial operations which build up and cancel valency position (3.2, 3.3). The NLC hear mode derivations used here continue to employ categorial operations (based on the cat feature of proplets), but show the construction of content essentially as a *cross-copying* of values, resulting in an order-free set of proplets suitable for storage and activation (retrieval) in a word bank.

Complementary to the evolution of a graphical representation for the hear mode, there remained the task of developing one for the speak mode. In a first attempt (NLC), we tried to define LA think and LA speak as separate but interacting, alternating mechanisms: a LA think navigation step from one proplet to a next triggered an LA speak application realizing as many surfaces as supported by the proplets traversed so far. The LA speak application in turn triggered a switch to LA think to continue the navigation. As an illustration, consider the following characterization of the discontinuous construction familiar from 2.3, 2.4, 3.3, and 5.1.

5.2 Schematic NLC production of Fido dug the bone up.

activated sequence				*realization*
i				
	V			
i.1		n		n
	V	N		
i.2	fv	n		n fv
	V	N		
i.3	fv	n	d	n fv d
	V	N	N	
i.4	fv	n	d nn	n fv d nn
	V	N	N	
i.5	fv de	n	d nn	n fv d nn de
	V	N	N	
i.6	fv de p	n	d nn	n fv d nn de p
	V	N	N	

The upper case letters V and N represent verb and noun proplets, respectively. The lower case letters represent abstract word form surfaces: n for name, fv for finite verb, d for determiner, nn for noun, de for discontinuous element, and p for period. The navigation order VNN produces surfaces in a different order, namely n fv d nn de p, representing Fido dug the bone up.

This worked reasonably well on paper, but was prohibitively complicated to program.

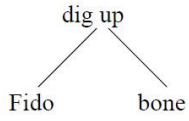
Therefore the idea of separate LA think and LA speak grammars had to go. Instead the language-dependent surface realization is now handled by embedding lexicalization rules into the sur slot of the proplets traversed by LAGo think. A LAGo think grammar with language-dependent lexicalization rules is called LAGo think-speak.

To graphically characterize the semantic relations traversed by LAGo think, we took an idea of Frege (1878) which has been lauded in the literature but, to the best of our knowledge, has not been taken up again until now. It assigns unique semantic interpretations to the different kinds of edges (lines) in a graph (instead of labeling edges).

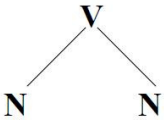
As an example, consider the following DBS graph analysis underlying the speak mode for producing Fido dug the bone up.:

5.3 DBS graph analysis of a discontinuous structure

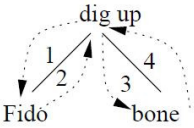
(i) *semantic relations graph (SRG)*



(ii) *signature*



(iii) *numbered arcs graph (NAG)*



(iv) *surface realization*

1	2	3	4	
Fido	dug	the_bone	up_	.
V/N	N/V	V\N	N\V	

The SRG and the signature show the semantic relations of structure using the edge "/" for the subject/predicate and the edge "\" for the object\predicate relation. The arrow numbering in the NAG is standard pre-order and is used in the *surface realization*. It shows that the surface Fido is realized from the goal node of transition 1, dug from 2, the_bone from 3, and up_. from 4.

Not unlike 4.2, the graphs (i-iii) in 5.3 have a superficial resemblance with a constituent structure. This may be welcome insofar as graphical representations have long been an essential part of linguistic intuitions. It should be clear, however, that the motivation behind the two kinds of graphs is different.

First, a constituent structure shows *dominance* and *precedence* in a language sign, while a DBS graph shows the compositional semantics of a content. Second, constituent structure is motivated by the movement and substitution tests of American Structuralism (Bloomfield 1933; Harris 1951), while the graphs of DBS are motivated by functor-argument and coordination relations at the elementary, phrasal, and clausal level of grammatical complexity.

6. Conclusion

Systematic long-term upscaling cannot be successful unless the overall approach is based on the correct method, ontology, and functional flow. The method of DBS allows to detect errors automatically,[24] locate them precisely in the software, and correct them permanently.[25] The ontology of DBS is agent-oriented, i.e., the external reality is treated as given and the scientific work concentrates on modeling the agent's recognition of and action in its external surroundings.[26] The functional flow of DBS reconstructs the speak and the hear mode as mappings between the external modality-dependent unanalyzed language surfaces and the content in the agent's memory.[27] By integrating method, ontology, and functional flow into the design of a talking robot, DBS ensures compatibility between components and has a broader empirical base than any other theory of language.

The basic motor driving central cognition is the changes in the agent's ecological niche, which compel cognition to constantly maintain and regain a state of balance. Modeling the agent's survival in an environment requires actual robots which interact with actual surroundings by means of their autonomous recognition and action interfaces. The environment may be artificial, e.g., a test course codesigned for a certain kind of robot, but it must be real.[28]

<References>

Ajdukiewicz, K. (1935) "Die syntaktische Konnexität", *Studia Philosophica* Vol. 1:1 – 27

Bar-Hillel, Y. (1964) *Language and Information. Selected Essays on Their Theory and Application,* Reading, Mass.: Addison-Wesley

Berwick, R. C. and A.S. Weinberg (1984) *The Grammatical Basis of Linguistic Performance: Language Use and Acquisition,* Cambridge, Mass.: MIT Press

---

24 They are revealed by parsing failures which occur when processing test lists or free text.

25 For the statistical method, error correction is limited to manual "post-editing", and must be done all over again from one text to the next. This is also the reason why statistical tagging cannot be used for spontaneous dialog involving nontrivial content.

26 Truth-conditional semantics bypasses the cognitive operations of the agent by constructing set-theoretical models which relate language signs directly to an abstraction of "the world."

27 The phrase structure grammars of nativism generate different expressions from the same start node S, without interfaces for recognition and action, without an agent-internal memory, and without a viable distinction between the speak and the hear mode.

28 Simulating the external environment on a standard computer is not an option. For example, in virtual reality the goal of modeling the action of raising a cup is a convincing *appearance*, e.g., getting the shadows right. The same action by a real robot, in contrast, requires building the actual gripping action of the artificial hand and movement of the artificial arm. Central concerns are not to break the cup and not to spill the liquid; a convincing handling of the shadows is left to nature.

Bloomfield, L. (1933) *Language,* New York: Holt, Rinehart, and Winston

Buszkowski, W. (1988) "Gaifman's theorem on categorial grammars revisited", *Studia Logica,* Vol. 47.1:23 – 33

Carbonell, J. G. and R. Joseph. (1986) "FrameKit+: a knowledge representation system", Carnegie Mellon U., Department of Computer Science

Choe, J. W. and R. Hausser (2006) "Handling Quantifiers in Database Semantics" in *Information Modeling and Knowledge Bases XVIII,* edited by M. Duzi et al. Amsterdam: IOS Press Ohmsha

CLaTR = Hausser, R. (2011) *Computational Linguistics and Talking Robots – Processing Content in Database Semantics,* Springer

CoL = Hausser, R. (1989) *Computation of Language,* Springer

FoCL = Hausser, R. (1999) *Foundations of Computational Linguistics, Human-Computer Communication in Natural Language, 3rd ed. 2013,* Springer

Hajičová, E. (2000) "Dependency-based underlying-structure tagging of a very large czech corpus", in: Special issue of TAL Journal, Grammaires de Dépendence/Dependency Grammars, p.57 – 78, Paris: Hermes

Harris, Z. (1951) *Method in Structural Linguistics,* Chicago: U. of Chicago Press

Hausser, R. (2003) "Reconstructing propositional calculus in Database Semantics", in H. Kangassalo et al. (eds.) (2003) *Information Modeling and Knowledge Bases XIV*, IOS Press Ohmsha, Amsterdam

Hausser, R. (2013) "Content-Based Retrieval in Database Semantics - A Theoretical Foundation for Practical NLP", in *Semantics in Data and Knowledge Bases*. LNCS 7693, Klaus-Dieter Schewe, Bernhard Thalheim (eds.), Springer

Kempson, R., W. Meyer-Viol, D. Gabbay. (2001) *Dynamic Syntax: The Flow of Language Understanding,* Wiley-Blackwell

Leśniewski, S. (1929) "Grundzüge eines neuen Systems der Grundlagen der Mathematik", Warsaw: *Fundamenta Mathematicae,* Vol. 14:1 – 81

NLC = Hausser, R. (2006) *A Computational Model of Natural Language Communication – Interpretation, Inference, and Production in Database Semantics,* Springer

NEWCAT = Hausser, R. (1986) *NEWCAT: Parsing Natural Language Using Left-Associative Grammar,* LNCS 231, Springer

Peters, S. and Ritchie, R. (1973) "On the generative power of transformational grammar", *Information and Control* Vol. 18:483 – 501

Saussure, F. de (1916/1972) *Cours de linguistique générale,* Édition critique préparée par Tullio de Mauro, Paris: Éditions Payot

SCG = Hausser, R. (1986) *Surface Compositional Grammar,* Munich: Wilhelm Fink Verlag

TCS = Hausser, R. (1992) "Complexity in left-associative grammar", *Theoretical Computer Science* Vol. 106.2:283 – 308

Tesnière, L. (1959) *Éléments de syntaxe structurale,* Editions Klincksieck, Paris

Zholkovskij, A. and Mel'chuk, A. (1965) "O vozmozhnom metode i instrumentax

semanticheskogo sinteza." (On a possible method and instruments for semantic synthesis.) Nauchno-texnicheskaja informacija. Retrieved May, 19, 2001, from the World Wide Web: http://www.neuvel.net/meaningtext.htm

Submitted on: September 1, 2015
Accepted on: October 26, 2015